Rethinking KL Divergence in RLHF: From Single Sample to Mini-Batch to Expectation

YiMing Liu[1]

[1]Email: letusgo126@126.com, From SYSU HCP Lab

March 2025

1 核心观点：有效样本量需足够大

在基于人类反馈的强化学习（RLHF）中，策略优化通常涉及约束策略 $\pi_\theta(a_t|s_t)$ 与参考策略 $\pi_{\text{ref}}(a_t|s_t)$ 之间的 KL 散度。KL 散度计算的是两个分布间的距离，在实践中成本高昂（分类需要在全词表上计算），因此通常利用蒙特卡洛方法基于采样估计来近似；每对 prompt 与 answer 作为一条轨迹，每次只保留采样类别，是分布的单次采样。并且当单次采样作为损失函数优化语言模型参数时，应该采样无偏的 KL 散度的梯度估计而非无偏的 KL 散度值估计。

基于采样梯度估计值优化 KL 散度分布，需要满足特定的蒙特卡洛条件才等价于直接优化 KL 散度分布：在策略参数 θ 保持近似不变（即 θ 在采样过程中更新幅度极小或不更新）的前提下，在 $\pi_\theta(a_t|s_t)$ 中采样得到足够大的样本量，才能保证轨迹采样的蒙特卡洛 KL 散度估计的梯度能够有效逼近真实 KL 散度分布对模型参数的梯度。形式化地，设 $\nabla_\theta \mathbb{D}_{\text{KL}}(\pi_\theta||\pi_{\text{ref}})$ 为 KL 散度分布的真实梯度，而 $\nabla_\theta \widehat{\mathbb{D}}_{\text{KL}}(\pi_\theta||\pi_{\text{ref}})$ 为基于有限采样样本估计的 KL 散度梯度的蒙特卡洛近似，则两者近似等价的条件是有效样本量足够大，具体来讲：

- 策略参数稳定（有效性）：在样本采样的过程中，策略参数 θ 应保持稳定或仅发生微小变化，以确保采样分布与当前策略分布的一致性。

- 采样样本量充足（充足性）：采样的样本数量必须足够大，以保证蒙特卡洛估计的收敛性。

下表总结了不同 RLHF 算法是否满足采样时策略参数 θ 的稳定以及有足够大的采样样本量：

	PPO	GRPO_on - policy	GRPO_off-policy	DPO
策略参数的稳定 (采样时 θ 相同或偏差极小)	✓	✓	✓	?
足够的采样样本量	✓	?	✓	✓

其中，“✓”表示满足条件，“×”表示不满足条件，“?”表示情况未知或依赖具体实现。该表格强调了在利用采样样本估计 KL 散度梯度时，策略参数的稳定性和样本量的充足性是至关重要的。

2 贡献

本文的主要学术贡献在于对基于 KL 散度的强化学习算法进行了深入的理论分析与实证探讨，具体体现在以下几个方面：

- **无偏估计 KL 散度梯度值而非 KL 散度值，估计有效的条件** 1）直接在全词表上按 KL 散度定义计算 KL 散度作为损失函数理论可行，但成本昂贵，实践中需要使用基于采样的方法，仅保存采样轨迹每个 token 的概率值。2）作为损失函数，更应该估计无偏低方差的 KL 散度梯度值而非 KL 散度值。在 appendix A中，本文证出了采样时 KL 散度梯度的解析表达式。估计有效的条件是有效样本量必须足够大：以稳定的策略采样到足够多的样本。

- **k1、k2、k3 估计** 1）在 section 3.2.2证明了 k1 估计在奖励函数中与 k2 估计直接作为损失函数之间的等价性，并在 section 4中证明了 k2 损失函数相较于 k3 损失函数的优越性。2）在 section 4.3证明了 GRPO 算法中 k3 损失函数向奖励函数中重参数化后奖励函数的表达式。3）举反例论证了 k3 值估计的方差也并非始终小于 k1 和 k2，反驳了现有文献中关于 k3 估计绝对优越性的观点（如 [8, 9]）。

KL 估计类型/是否适合作为	reward 函数中	直接作为 loss
k1	✓	×
k2	×	✓
k3	×	有偏
变换并转移到 reward 函数中的 k3 section 4.3	有偏	×

- **推导并阐明 PPO 算法的奖励函数：**在 section 3.4，推导了 PPO 算法的原始奖励函数公式，对现有文献中可能存在的模糊表述 [6, 11, 3] 进行了澄清。PPO 奖励函数有效是因为实践中恰好满足了有效样本量足够大的条件。
- **修正并扩展 GRPO 算法的理论与实现：** GRPO 原论文的算法 [9] 仅在 on-policy 下有效，但其在 off-policy 下是近似或为不正确的。针对此问题，本文在 section 5中提供了理论修正，并给出了具体的代码实现（见[https://github.com/OpenRLHF/OpenRLHF/pull/840]），扩展了 GRPO 算法的适用范围。

3 KL 梯度和 PPO reward 函数推导

3.1 RLHF 目标优化函数建模

基于人类反馈的强化学习（RLHF）可建模为以下优化问题：

$$\arg\max_{\theta} J = \underbrace{\mathbb{E}_{x\sim D, y\sim\pi_\theta(\cdot|x)}\left[r(x,y)\right]}_{\text{奖励最大化项}} - \beta \underbrace{\left[\mathbb{D}_{\mathrm{KL}}\left(\pi_\theta(a_t|s_t) \,||\, \pi_{\mathrm{ref}}(a_t|s_t)\right)\right]}_{\text{策略约束项}} \tag{1}$$

$$s_t \sim d^\pi(s) := \mathbb{E}_{x\sim\mathcal{D}}\left[\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\left(\sum_{t=0}^{T-1}\mathbb{I}\{s_t = s\}\right)\right], a_t \sim \pi_\theta(\cdot|x) \tag{2}$$

其中 x 为输入 prompt，y 为输出 answer。π_θ 为 LLM 在参数 θ 时的输出概率，是变量，θ 为待参数优化。π_{ref} 为参考模型，不是变量，参数不优化。β 为温度系数，控制策略偏离参考模型 π_{ref} 的程度。该目标函数包含两个关键部分：奖励期望的最大化和 KL 散度约束。

3.2 RLHF 目标优化函数分解与具体实现

结合奖励期望的最大化函数的损失项 $\mathcal{L}_R(\theta)$ 和 KL 散度约束项 $\mathcal{L}_{\mathrm{KL}}(\pi_\theta, \pi_{\mathrm{ref}})$，完整损失函数可表示为：

$$L_{\mathrm{total}} = -\left(\mathcal{L}_R(\theta) - \beta\mathcal{L}_{\mathrm{KL}}(\pi_\theta, \pi_{\mathrm{ref}})\right) \tag{3}$$

3.2.1 奖励最大化的损失函数

奖励项的损失函数为：

$$-\mathcal{L}_R(\theta) = \mathbb{E}_{x\sim D, y\sim\pi_\theta(\cdot|x)}\left[r(x,y)\right] = \mathbb{E}_{x\sim D}\sum_{a_t\in Vocab}\left[\pi_\theta(a_t|s_t)r(x,y)\right] \tag{4}$$

求梯度后进行等价变换：

$$\begin{aligned}
-\nabla_\theta\mathcal{L}_R(\theta) &= \nabla_\theta\mathbb{E}_{x\sim D}\sum_{a_t\in Vocab}\left[\pi_\theta(a_t|s_t)\cdot r(x,y)\right] \\
&= \mathbb{E}_{x\sim D}\sum_{a_t\in Vocab}\left[\nabla_\theta\pi_\theta(a_t|s_t)\cdot r(x,y)\right] \\
&= \mathbb{E}_{x\sim D}\sum_{a_t\in Vocab}\left[\pi_\theta(a_t|s_t)\frac{\nabla_\theta\pi_\theta(a_t|s_t)}{\pi_\theta(a_t|s_t)}\cdot r(x,y)\right] \\
&= \mathbb{E}_{x\sim D}\sum_{a_t\in Vocab}\left[\pi_\theta(a_t|s_t)\nabla_\theta\log\pi_\theta(a_t|s_t)\cdot r(x,y)\right] \\
&= \mathbb{E}_{x\sim D, y\sim\pi_\theta(\cdot|x)}\left[\nabla_\theta\log\pi_\theta(a_t|s_t)\cdot r(x,y)\right]
\end{aligned} \tag{5}$$

根据梯度，可以将奖励项的损失函数重写为更加常见的形式：

$$-\mathcal{L}_R(\theta) = \mathbb{E}_{x\sim D, y\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\left[r(x,y)\log\pi_\theta(a_t|s_t)\right] \tag{6}$$

其中 θ_{old} 为采样时的模型参数，不是变量；θ 为 RL 训练时的模型参数，π_θ 是变量。在 on-policy 时，θ_{old} 和 θ 二者在数值上相等。

实际计算中，由于 reward model 的限制，对全词表的 a_t 计算 $r(x,y)$ 是几乎不可能实现的。通过蒙特卡洛估计采样并计算 $r(x,y)$ 在 y 上的期望：

$$-\mathcal{L}_R(\theta) = \frac{1}{M\cdot N}\sum_{i=1}^{M}\sum_{j=1}^{N}\pi_{\theta_{\text{old}}}(a_t^j|s_t^i)\left[r(x,y)\log\pi_\theta(a_t^j|s_t^i)\right] \tag{7}$$

蒙特卡洛算法有效的要求是有效样本量足够大。

3.2.2 KL 惩罚的损失函数

KL 惩罚定义可以写为：

$$\text{KL}(\pi_\theta(a_t|s_t)\|\pi_{\text{ref}}(a_t|s_t)) = \mathbb{E}_{x\sim D, y\sim\pi_\theta(\cdot|x)}\left[\log\frac{\pi_\theta(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)}\right] \tag{8}$$

KL 惩罚梯度的推导具体推导见 eq. (53)，结果为：

$$-\nabla_\theta\text{KL}(\pi_\theta(a_t|s_t)\|\pi_{\text{ref}}(a_t|s_t)) = -\mathbb{E}_{x\sim D, y\sim\pi_\theta(\cdot|x)}\left[\log\frac{\pi_\theta(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)}\cdot\nabla_\theta\log\pi_\theta(a_t|s_t)\right] \tag{9}$$

在 reward 函数中 k1 估计的损失函数形式 因此损失函数可推导为:

$$\mathcal{L}_{\mathrm{KL}}(\pi_\theta, \pi_{\mathrm{ref}}) = \mathbb{E}_{x\sim D, y\sim\pi_{\theta_{\mathrm{old}}}(\cdot|x)}\left[\log\frac{\pi_{\theta_{\mathrm{old}}}(a_t|s_t)}{\pi_{\mathrm{ref}}(a_t|s_t)}\cdot\log\pi_\theta(a_t|s_t)\right] \tag{10}$$

这个损失函数既可以单独使用，更多是如下文 eq. (20) 所述像 PPO 一样重参数化到 reward 函数中。该损失函数需要对全词表的 y 计算期望，可以在全词表上直接计算分布，但实践上更多用蒙特卡洛采样估计样本梯度优化分布，需要有效样本量足够大。

k2 损失函数形式 如 appendix A.2证明:

k_2 损失函数为:

$$\mathcal{L}_{k_2}(\theta) = \frac{1}{2}\mathbb{E}_{x\sim D, y\sim\pi_{\theta_{\mathrm{old}}}(\cdot|x)}\left[\left(\log\frac{\pi_\theta(a_t|s_t)}{\pi_{\mathrm{ref}}(a_t|s_t)}\right)^2\right] \tag{11}$$

进一步验证，求 $\mathcal{L}_{k_2}(\theta)$ 损失函数的梯度:

$$-\nabla_\theta\mathcal{L}_{k_2}(\theta) = -\frac{1}{2}\mathbb{E}_{x\sim D, y\sim\pi_{\theta_{\mathrm{old}}}(\cdot|x)}\left[\nabla_\theta\left(\log\frac{\pi_\theta(a_t|s_t)}{\pi_{\mathrm{ref}}(a_t|s_t)}\right)^2\right] \tag{12}$$

$$= -\mathbb{E}_{x\sim D, y\sim\pi_{\theta_{\mathrm{old}}}(\cdot|x)}\left[\log\frac{\pi_{\theta_{\mathrm{old}}}(a_t|s_t)}{\pi_{\mathrm{ref}}(a_t|s_t)}\cdot\nabla_\theta\log\pi_\theta(a_t|s_t)\right] \tag{13}$$

得出结论在 reward 函数中 k1 估计的损失函数等价于 K2 损失函数:

$$\nabla_\theta\mathcal{L}_{k_2}(\theta) = \nabla_\theta\mathrm{KL}(\pi_\theta(a_t|s_t)\|\pi_{\mathrm{ref}}(a_t|s_t)) \tag{14}$$

进而得出

$$\mathcal{L}_{\mathrm{KL}}(\pi_\theta, \pi_{\mathrm{ref}}) = \nabla_\theta\mathcal{L}_{k_2}(\theta) \tag{15}$$

3.3 RLHF 目标优化函数具体实现的合并与 PPO reward 函数推导

两个组成部分:

1. **奖励项梯度**:

$$\nabla_\theta\mathcal{L}_R(\theta) = \mathbb{E}_{x\sim D, y\sim\pi_{\theta_{\mathrm{old}}}(\cdot|x)}\left[r(x,y)\nabla_\theta\log\pi_\theta(a_t|s_t)\right] \tag{16}$$

2. **KL 散度项梯度**（具体推导见引理 eq. (53)）:

$$-\nabla_\theta\mathcal{L}_{\mathrm{KL}} = -\mathbb{E}_{x\sim D, y\sim\pi_{\theta_{\mathrm{old}}}(\cdot|x)}\left[\log\frac{\pi_{\theta_{\mathrm{old}}}(a_t|s_t)}{\pi_{\mathrm{ref}}(a_t|s_t)}\cdot\nabla_\theta\log\pi_\theta(a_t|s_t)\right] \tag{17}$$

将二者结合可得总梯度:

$$
\begin{aligned}
-\nabla_\theta L_{total} &= \nabla_\theta \mathcal{L}_R - \beta \nabla_\theta \mathcal{L}_{\mathrm{KL}} \\
&= \mathbb{E}_{x\sim D, y\sim \pi_{\theta_{\mathrm{old}}}(\cdot|x)} \left[\left(r(x,y) - \beta \log \frac{\pi_{\theta_{\mathrm{old}}}(a_t|s_t)}{\pi_{\mathrm{ref}}(a_t|s_t)} \right) \nabla_\theta \log \pi_\theta(a_t|s_t) \right]
\end{aligned} \tag{18}
$$

3.4 等效优化目标的构造与 PPO 算法的 reward 函数

由公式 eq. (18) 可以重新推导并定义新的优化目标:

$$
\arg\max_\theta J' = \mathbb{E}_{x\sim D, y\sim \pi_\theta(\cdot|x)} \left[r(x,y) - \beta \log \frac{\pi_{\theta_{\mathrm{old}}}(a_t|s_t)}{\pi_{\mathrm{ref}}(a_t|s_t)} \right] \tag{19}
$$

该形式将 KL 正则项内化为奖励函数的组成部分，从而实现了优化目标的重新参数化。

提取出强化学习整个奖励部分，容易得出与 PPO 算法相同的奖励函数为:

$$
\tilde{r}(a_t|s_t) = r(x,y) - \beta \log \frac{\pi_{\theta_{\mathrm{old}}}(a_t|s_t)}{\pi_{\mathrm{ref}}(a_t|s_t)} \tag{20}
$$

这个时候由于 $r(x,y)$ 的限制并且 KL 项合并到一起了，整个式子使用蒙特卡洛估计，所以有效样本量一定要足够大。

4 优化 KL 散度方法分析

4.1 基于采样梯度估计分布梯度

KL 梯度的定义为:

$$
-\nabla_\theta \mathcal{L}_{\mathrm{KL}} = -\mathbb{E}_{x\sim D, y\sim \pi_{\theta_{\mathrm{old}}}(\cdot|x)} \left[\log \frac{\pi_{\theta_{\mathrm{old}}}(a_t|s_t)}{\pi_{\mathrm{ref}}(a_t|s_t)} \cdot \nabla_\theta \log \pi_\theta(a_t|s_t) \right] \tag{21}
$$

该式对全词表的 y 求期望。在代码工程实践中，因为显存和计算量等限制，直接计算全词表的 y 理论可以，但成本昂贵；往往会用蒙特卡洛采样等手段，采样 Mini-batch 后求对模型参数的梯度并进行模型参数的更新。Mini-batch 是两种最极端的情况的中间状态，有效样本量为 1 何有效样本量趋于无穷（期望)。我们不仅得考虑在期望上的推导，还得考虑单样本的情况，进而评估 Mini-batch 下模型优化的状态。

分别具体讨论基于全词表和基于采样的做法:

全词表 LLM 本质上是一个分类器，它的类别终归是有限的（1w∼3w），遍历完整词表能基于 KL 散度定义计算得到 KL 分布，也等效为遍历完整词表直接计算的 KL 散度梯度：

$$-\nabla_\theta \mathcal{L}_{\mathrm{KL}} = -\mathbb{E}_{x\sim D} \sum_{i=1}^{vocab\_size} \pi_{\theta_{\mathrm{old}}}(a_t^i|s_t) \left[\log \frac{\pi_{\theta_{\mathrm{old}}}(a_t^i|s_t)}{\pi_{\mathrm{ref}}(a_t^i|s_t)} \cdot \nabla_\theta \log \pi_\theta(a_t^i|s_t)\right] \tag{22}$$

这种方法的好处是即便 batch_size=1，单样本，计算的也是完整的 KL 散度和精确的 KL 散度梯度。

基于蒙特卡洛采样 因为词表过大，在更多的工程实践中，人们往往会用蒙特卡洛采样的手段，通过在有限样本上近似估计 KL 散度梯度，再进行 Mini-batch 更新：

$$-\nabla_\theta \mathcal{L}_{\mathrm{KL}} = -\frac{1}{N}\mathbb{E}_{x\sim D} \sum_{i=1}^{N} \pi_{\theta_{\mathrm{old}}}(a_t^i|s_t) \left[\log \frac{\pi_{\theta_{\mathrm{old}}}(a_t^i|s_t)}{\pi_{\mathrm{ref}}(a_t^i|s_t)} \cdot \nabla_\theta \log \pi_\theta(a_t^i|s_t)\right] \tag{23}$$

其中 N 是采样次数。理论上，只要采样数量足够大，采样梯度也能够很好近似 KL 分布的梯度。这种方案每次只要记录下采样轨迹的 y_i 的单类别概率值，显存消耗少，但它必须满足有效样本量足够大才等效于在全词表上精确计算的 KL 散度及梯度。满足有效样本量足够大并非易事，采样样本量足够大且参数不能变化，这意味着采样次数多，更新次数少，训练效率不高。在工程实践中，往往都是 Mini-batch，它是单样本和期望上的一个中间状态，即便如此，仍需精心设计，使其在高效的同时，满足“有效样本量”足够大的采样条件。

4.2 不同形式 KL 估计在 Mini-batch 下的性质

KL 估计主要定义为 k1、k2、k3[8]，下将介绍不同形式 KL 估计的在单样本和期望上的采样梯度，它们用来近似 KL 分布梯度并优化 KL 分布，下面是具体做法介绍和性质。

4.2.1 何种 KL 估计适合直接作为损失函数

k1 估计 k1 直接作为 loss 对应的损失函数形式为:

$$\begin{aligned}\mathcal{L}_{\text{KL\_k1 loss}}(\pi_\theta, \pi_{\text{ref}}) &= \mathbb{E}_{x\sim D, y\sim\pi_{\theta_{\text{old}}}(\cdot|x)}(\log \pi_\theta(a_t|s_t) - \log \pi_{\text{ref}}(a_t|s_t)) \\ &= \mathbb{E}_{x\sim D} \sum_{i=1}^{batch\_size} \pi_\theta(a_t^i|s_t)(\log \pi_\theta(a_t^i|s_t) - \log \pi_{\text{ref}}(a_t^i|s_t))\end{aligned} \tag{24}$$

其梯度表达式为:

$$-\nabla_\theta \mathcal{L}_{\text{KL\_k1 loss}} = -\mathbb{E}_{x\sim D} \sum_{i=1}^{batch\_size} \pi_\theta(a_t^i|s_t) \nabla_\theta \log \pi_\theta(a_t^i|s_t) \tag{25}$$

- **单样本**:

$$-\nabla_\theta \mathcal{L}_{\text{KL\_k1 loss}} = -\mathbb{E}_{x\sim D} \pi_\theta(a_t^i|s_t)(\nabla_\theta \log \pi_\theta(a_t^i|s_t)) \tag{26}$$

  该梯度恒指向降低当前策略概率的方向，采样到某个样本就降低其生成概率。会导致策略网络的概率输出持续衰减，最终引发模型崩溃。

- **期望**: 对梯度求期望:

$$\begin{aligned}-\nabla_\theta \mathcal{L}_{\text{KL\_k1 loss}} &= -\mathbb{E}_{x\sim D} \sum_{i=1}^{\infty} \pi_\theta(a_t^i|s_t) \nabla_\theta \log \pi_\theta(a_t^i|s_t) \\ &= -\mathbb{E}_{x\sim D} \sum_{i=1}^{\infty} \pi_\theta(a_t^i|s_t) \frac{\nabla_\theta \pi_\theta(a_t^i|s_t)}{\pi_\theta(a_t^i|s_t)} \\ &= -\mathbb{E}_{x\sim D} \sum_{i=1}^{\infty} \nabla_\theta \pi_\theta(a_t^i|s_t) \\ &= -\mathbb{E}_{x\sim D} \nabla_\theta \sum_{i=1}^{\infty} \pi_\theta(a_t^i|s_t) \\ &= -\mathbb{E}_{x\sim D} \nabla_\theta 1 = 0\end{aligned} \tag{27}$$

  在期望上模型梯度为 0，就是说有效样本量特别大的时候，k1 估计直接作为 loss 对模型更新几乎无影响。

- 综上，无论哪种情况，都不是在估计 KL 散度分布的梯度。尤其在有效样本量比较小且 KL 惩罚项系数 β 比较大时，模型大概率会崩溃。

k2 估计

$$
\begin{aligned}
-\nabla_\theta \mathcal{L}_{\text{KL\_k2 loss}} &= -\mathbb{E}_{x\sim D, y\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\left[\log\frac{\pi_{\theta_{\text{old}}}(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)}\cdot\nabla_\theta\log\pi_\theta(a_t|s_t)\right] \\
&= -\mathbb{E}_{x\sim D}\sum_{i=1}^{batch\_size}\pi_{\theta_{\text{old}}}(a_t^i|s_t)\left[\log\frac{\pi_{\theta_{\text{old}}}(a_t^i|s_t)}{\pi_{\text{ref}}(a_t^i|s_t)}\cdot\nabla_\theta\log\pi_\theta(a_t^i|s_t)\right]
\end{aligned}
\tag{28}
$$

期望 如 eq. (61) 所示，推导出了 k2 估计直接作为损失函数并证明在期望上等效于计算 KL 散度的梯度，进而优化 KL 散度。有效样本量足够大时，能够很好起到限制策略模型和参考模型的 KL 距离的作用，防止策略模型崩溃。

单样本 单样本估计的梯度如下所示：

$$
\begin{aligned}
-\nabla_\theta \mathcal{L}_{\text{KL\_k2 loss}} &= -\mathbb{E}_{x\sim D}\pi_{\theta_{\text{old}}}(a_t^i|s_t)\left[\log\frac{\pi_{\theta_{\text{old}}}(a_t^i|s_t)}{\pi_{\text{ref}}(a_t^i|s_t)}\cdot\nabla_\theta\log\pi_\theta(a_t^i|s_t)\right] \\
&= \mathbb{E}_{x\sim D}\pi_{\theta_{\text{old}}}(a_t^i|s_t)\left[(\log\pi_{\text{ref}}(a_t^i|s_t)-\log\pi_{\theta_{\text{old}}}(a_t^i|s_t))\cdot\nabla_\theta\log\pi_\theta(a_t^i|s_t)\right]
\end{aligned}
\tag{29}
$$

当 $\log\pi_{\text{ref}}(a_t^i|s_t) > \log\pi_{\theta_{\text{old}}}(a_t^i|s_t)$,$-\nabla_\theta\mathcal{L}_{\text{KL}} < 0$, 会降低 $\log\pi_{\theta(a_t^i|s_t)}$ 生成概率。反之亦然。单样本尽管没有起到限制 KL 的作用，但是它对这个样本的类别 y_i 起到了一定限制作用，防止它偏离参考模型过远。所以在小学习率和小样本下，再配合 eq. (20) 中的 β 小系数，工作得也不错。

但必须强调的是，在单样本或有效样本量很低时，使用大学习率也有可能无法限制好 KL 距离，约束好策略模型的空间，最终造成模型崩溃。对于 on-policy 算法，比如 GRPO，[5] 中提到其计算集群包括 2048 张 H800 计算卡，能够把 batch_size 开到 1k∼2k, 再配合梯度累积技术，可能不会碰到有效样本量比较小的问题，能够很好优化 KL 分布。但个人研究者手中可能没有那么多显卡，他们使用 on-policy 算法时 batch_size 可能非常小，导致蒙特卡洛条件不成立。建议资源有限时更多结合梯度累积技术，或者使用一些结合重要性采样的 off-policy 算法（RF++/GRPO_off-policysection 5），这些算法 rollout 的 batch_size 会相对可观，能够满足更多的有效样本。

k3 估计是有偏的 k3 作为损失函数如下所示：

$$
\mathcal{L}_{\text{KL\_k3 loss}} = \mathbb{E}_{x\sim D, y\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\left(\frac{\pi_{\text{ref}}(a_t|s_t)}{\pi_\theta(a_t|s_t)} - \log\frac{\pi_{\text{ref}}(a_t|s_t)}{\pi_\theta(a_t|s_t)} - 1\right) \tag{30}
$$

对应的梯度表达式揭示其近似"k2 估计作为损失函数" 的本质：

$$-\nabla_\theta \mathcal{L}_{\text{KL\_k3 loss}} = \mathbb{E}_{x\sim D, y\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\left[\left(\frac{\pi_{\text{ref}}(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} - 1\right)\nabla_\theta \log \pi_\theta(a_t|s_t)\right] \tag{31}$$

令 $x = \frac{\pi_{\text{ref}}(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$，"k2 估计作为损失函数" 与 "k3 估计作为损失函数" 的梯度可对比表示为：

$$\text{k2 估计作为损失函数的梯度} : \log x \cdot \nabla_\theta \log \pi_\theta \tag{32}$$

$$\text{k3 估计作为损失函数的梯度} : (x-1) \cdot \nabla_\theta \log \pi_\theta \tag{33}$$

在策略邻域 ($x \approx 1$) 进行泰勒展开时，$\log x \approx x - 1$，k2 梯度是无偏的，此时 k3 梯度构成 k2 梯度的线性近似。但这种近似具有两个关键缺陷：1. **有偏性**：当策略显著偏离参考策略 (x 远离 1，训练后期 π_{ref} 离 $\pi_{\theta_{\text{old}}}$ 较远，尤其是 $\pi_{\text{ref}} >> \pi_{\theta_{\text{old}}}$) 时 (模型输出的类别为低概率范围)，近似误差呈非线性增长，可视化见 fig. 2。2. **非对称性**：$x-1$ 对 $\pi_{\theta_{\text{old}}} > \pi_{\text{ref}}$ 和 $\pi_{\theta_{\text{old}}} < \pi_{\text{ref}}$ 的响应特性不对称。所以"k3 估计作为损失函数" 仅仅是"k2 估计作为损失函数" 的近似，可以明确证明，k3 loss 比 k2 loss 更好！实验中也容易观察到使用 k3 loss 的 GRPO 算法比用 k2 loss 的方差波动更大。

4.2.2 何种 KL 估计适合放在奖励函数中

仅特定形式的 KL 估计适合作为 PPO 奖励函数中的惩罚部分。

根据理论分析，k1 估计是适用的，其表达式如 eq. (20) 所示，并且在 eq. (10) 有着推导。它与 k2 估计直接作为损失函数是等价的 eq. (14)，它的性质也完全相同。

然而，k2 和 k3 估计无论如何均不适合作为惩罚项，原因如下：考虑 k2 和 k3 的估计值 kl 具有恒正特性，其梯度方向始终满足：

$$-\nabla_\theta \text{KL}(\pi_\theta \| \pi_{\text{ref}}) = -\mathbb{E}_{x\sim D, y\sim\pi_{\theta_{\text{old}}}(\cdot|x)} kl \nabla_\theta \log \pi_\theta(a_t|s_t) \tag{34}$$

此时无论 $\pi_\theta(a_t|s_t)$ 与 $\pi_{\text{ref}}(a_t|s_t)$ 的概率分布关系如何，$-\nabla_\theta \text{KL}(\pi_\theta \| \pi_{\text{ref}})$ 衡小于 0，梯度更新方向都会强制降低当前策略 π_θ 生成任意动作 y 的概率。这种单向的惩罚机制会导致模型崩溃，无论是单样本采样还是期望。

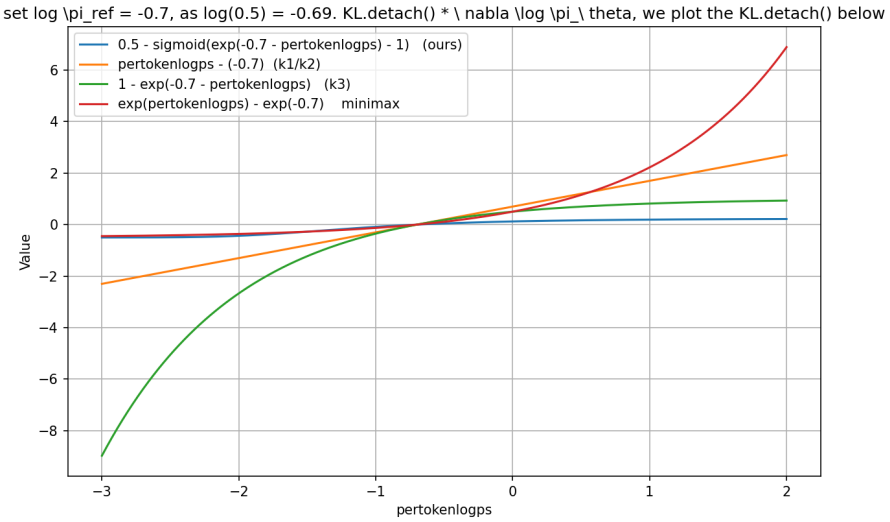

图 1: k1/k2 和 k3 及 MiniMax-01[4] 实现的 KL 梯度系数项对比，代码见 appendix F，相比 k1/k2,k3 在模型相比参考模型概率低很多时，数值会发生指数级变化，造成不稳定。(图中设定 $\log \pi_{\text{ref}} = -0.7$，此时概率 $\pi_{\text{ref}} \approx 0.5$)

4.3 Reward 函数中的 KL 估计和 KL 估计直接作为损失函数是可以相互转换的

k1 估计既可以如 eq. (20) 的 Reward 中的 k1 估计部分 $-\beta \log \frac{\pi_{\theta_{\text{old}}}(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)}$ 重参数化到 reward 中，也可以如 eq. (10) 直接作为作为损失函数，同时还完全等价于 k2 估计直接作为损失函数 eq. (61)。

结合 eq. (16) 和 eq. (31)，可以得出把 “k3 直接作为损失函数” 重参数化到 reward 函数中，易得 GRPO_off-policysection 5的 reward 函数：

$$\tilde{r}(a_t|s_t) = r(x,y) - \beta \left(1 - \frac{\pi_{\text{ref}}(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\right) \tag{35}$$

5 GRPO_off-policy

GRPO_on-policy 在个人显卡比较少的时候，即便开启梯度累积，batch_size 都比较小，有效样本量小。解决的办法是引入 PPO-2 的重要性采样和 clip，变成 PPO 一样的 off-policy 算法，就可以增大 rollout_batch，rollout_batch

图 2: 不同 KL 的实验，GRPO 算法，batch=12，group_size=6, 基于 x_r1 框架 [1]，$\beta = 0.04$

是在模型参数不变的情况下分批采样出来的，能够在资源有限的前提下做到有效样本量足够大。

GRPO 原论文中给出的公式为：

$$\begin{aligned}\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(O|q)] \\ \frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left\{\min\left[\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q,o_{i,<t})}\hat{A}_{i,t},\right.\right. \\ \left.\left.\text{clip}\left(\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q,o_{i,<t})}, 1-\varepsilon, 1+\varepsilon\right)\hat{A}_{i,t}\right] - \beta \mathrm{D}_{KL}[\pi_\theta||\pi_{\text{ref}}]\right\},\end{aligned} \tag{36}$$

参考 [13] 可以将上式的简化表示为：

$$\mathbb{E}_{\pi_{\theta_{\text{old}}}}\left[\frac{\pi_\theta}{\pi_{\theta_{\text{old}}}}A - KL(\pi_\theta, \pi_{\text{ref}})\right] = \mathbb{E}_{\pi_{\theta_{\text{old}}}}\left[\frac{\pi_\theta}{\pi_{\theta_{\text{old}}}}A - \mathbb{E}_{\pi_\theta}(\log\pi_\theta - \log\pi_{\text{ref}})\right] \tag{37}$$

因此，当将 KL 项用作损失时，按公式表达的意思是在期望上计算 KL 散度。

目前主流代码库 TRL[12]、OpenRLHF[2] 和 Verl[10] 代码中的 “compute_approx_kl” 函数的具体实现则是：

$$\mathbb{E}_{\pi_{\theta_{\text{old}}}}\left[\frac{\pi_\theta}{\pi_{\theta_{\text{old}}}}A - \mathbb{E}_{\pi_{\theta_{\text{old}}}}(\log\pi_\theta - \log\pi_{\text{ref}})\right] \tag{38}$$

由于我们实际上使用 $\pi_{\theta_{\text{old}}}$ 进行采样，并用 π_θ 去更新，除非是完全 on-policy 的状况，否则会导致丢失 π_θ 的导数信息，原始实现仅在 on-policy 或保留 “完整词汇表（vocab_size）” 并在整个词表上按 KL 散度定义计算时成立，像 PPO 那种 off-policy 的话 GRPO 的原论文写法是有问题的。

除了使用全词表、on-policy 外，我们更常用的 off-policy 时的 GRPO 必须修正，而修正的方法就是把 KL 散度的梯度估计也加上重要性采样和 PPO2-clip，合并同类项后，等同于把 KL 散度像 PPO 一样藏到 Reward 里面去：

在 reward 函数中的 k1 估计/ “k3 估计直接作为损失函数”：$\hat{R}_{i,t} = (\hat{A}_{i,t} - \beta\log\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\text{ref}}(o_{i,t}|q,o_{i,<t})})$

“k3 估计直接作为损失函数” 转移到 reward 函数中（有偏）：$\hat{R}_{i,t} = \left(\hat{A}_{i,t} - \beta\left(1 - \frac{\pi_{\text{ref}}(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\right)\right)$

$$
\begin{aligned}
\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(O|q)] \\
\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|} \\
\left\{\min\left[\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q,o_{i,<t})}\hat{R}_{i,t},\right.\right. \\
\left.\left.\text{clip}\left(\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q,o_{i,<t})}, 1-\varepsilon, 1+\varepsilon\right)\hat{R}_{i,t}\right]\right\}.
\end{aligned}
\tag{39}
$$

相关代码已经提交给 OpenRLHF。从这个角度，GRPO 独特性可能仅存在于 Group Norm 了。

总结起来，GRPO 有三种 KL 实现方法：1) 在完整词表按定义计算 KL，消耗巨大算力和显存，2)on-policy 堆硬件大显存，实现大 batch_size，3)GRPO_off-policy，综合成本效果最优。

5.1 Group Norm 的改进

Group Norm 是 GRPO 独有的: 同一个 prompt 生成 G 个 answer，用 reward model 打分后得到 group 的奖励列表 $\mathbf{r} = \{r_1, r_2, \cdots, r_G\}$, 然后对 group 的 reward 进行如下归一化得到优势列表 $\hat{A}_{i,t}$:

$$
\hat{A}_{i,t} = \widetilde{r}_i = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}, \tag{40}
$$

但 GRPO 在非 rule-based reward 上往往容易崩溃，这大概率是因为其 Group Norm 中/std() 的操作导致的: 比如 reward_list = [0.99999,1.00001,0.99999,1.00001], Adv_list = [-0.8660, 0.8660, -0.8660, 0.8660]。也就是说，方差非常小，完全可能是数值误差，误差会被 Group Norm 放大到一个很夸张的程度。GRPO 这种峰值奖励特性既给它带来了快速收敛性，也带来了不稳定性。

想要避免出现极端情况，建议对/std 进行 clip 操作，为此我们引入 clip_std 函数：$clip\_std() = max(min(std(), max), min)$, 此时优势计算函数变为:

$$
\hat{A}_{i,t} = \widetilde{r}_i = \frac{r_i - \text{mean}(\mathbf{r})}{\text{clip\_std}(\mathbf{r})}, \tag{41}
$$

PS：当 reward model 的打分在 0.0-1.0 间时，Group Norm 的 reward_list 的 std() 计算到的方差一定是 <1 的，证明在 appendix E，起

到的作用一定是增大区分度。并且 reward_list 方差越小，放大程度越大，/std 为区分度增加越显著。

6 DPO 的局限性分析

尽管直接偏好优化（DPO）在实践中取得了显著的成功，但其理论基础与实际应用之间仍存在一定的差距，尤其是在优化分布的层面。

DPO 的理论目标是优化公式 eq. (67) 所定义的**分布**，但在实际操作中，如公式 eq. (66) 所示，这一优化过程是通过对**采样样本**实现的。根据本文的核心观点，利用基于采样估计的梯度来近似分布梯度，并优化分布需要满足蒙特卡洛条件，即有效样本量足够大。然而，DPO 作为一种 offline 算法，在满足这一条件方面面临固有的挑战，很难满足采样时策略函数稳定的条件。

具体而言，1) 现有研究中可能采用来自外部数据集（如 GPT-4 收集的数据）或其他模型生成的数据来训练 DPO 模型。这些数据的分布与当前待优化模型的分布可能存在显著差异。2) 其次，即使采用同一模型进行采样、标注数据并训练该模型，也存在潜在问题。在 DPO 训练过程中，模型参数会经历多次更新，导致其模型分布发生变化，与数据采样时的分布不再一致。

文章 [14] 的观察结果进一步佐证了这一问题：使用相同模型采样的数据训练该模型时，chosen reward 可能会持续上升；而使用其他模型采样的数据训练当前模型时，chosen reward 则可能下降。这暗示了 DPO 在利用异构数据或历史数据来优化当前策略分布时可能存在困难。

为了缓解上述问题，一些潜在的解决方案包括转向更接近 online 的算法模式，例如，1）循环进行采样、标注和训练的过程；2）在 DPO 损失中为正样本引入额外的 SFT loss（极端情况，拿数据正样本 SFT 重训练要 DPO 训练的 LLM 模型，LLM 模型采样分布和训练数据分布不就一致了），从而在一定程度上近似满足蒙特卡洛条件。

然而，DPO 的核心优势在于其 offline 特性，能够高效地利用大规模预先标注的数据。如果为了满足蒙特卡洛条件而牺牲其 offline 的便利性，改为接近 online 算法，可能会导致 DPO 的失去核心优势，并可能引入额外的复杂性。

综上所述，DPO 的最终目标是优化分布，但在实践中，其优化过程依赖于有限的 Mini-batch 样本，且数据样本分布和此时策略模型采样的样本

分布很难保持一直。为了实现最佳的优化效果，训练数据与模型之间必须满足或近似满足蒙特卡洛条件，如何有效地利用 offline 数据来优化分布，尽可能地满足蒙特卡洛条件，是 DPO 进一步改进的关键挑战。

参考文献

[1] dhcode cpp. X-r1, 2025.

[2] Jian Hu, Xibin Wu, Zilin Zhu, Weixun Wang, Dehao Zhang, Yu Cao, et al. Openrlhf: An easy-to-use, scalable and high-performance rlhf framework. *arXiv preprint arXiv:2405.11143*, 2024.

[3] Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019.

[4] Aonian Li, Bangwei Gong, Bo Yang, Boji Shan, Chang Liu, Cheng Zhu, Chunhao Zhang, Congchao Guo, Da Chen, Dong Li, et al. Minimax-01: Scaling foundation models with lightning attention. *arXiv preprint arXiv:2501.08313*, 2025.

[5] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[6] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[7] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.

[8] John Schulman. Approximating kl divergence, 2020.

[9] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[10] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv:2409.19256*, 2024.

[11] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021, 2020.

[12] Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. Trl: Transformer reinforcement learning. `https://github.com/huggingface/trl`, 2020.

[13] Hongyu Zang. The critical implementation detail of kl loss in grpo, 2025. Notion Blog.

[14] 蜡笔小熊猫. 为什么 dpo 的 chosen reward 会和 reject reward 一起降低？, 2024.

A 分类模型 KL 散度梯度推导

A.1 KL 散度的定义

KL 散度衡量两个离散概率分布 π_θ 和 π_{ref} 之间的差异，定义为：

$$\begin{aligned}\text{KL}(\pi_\theta(a_t|s_t)\|\pi_{\text{ref}}(a_t|s_t)) &= \mathbb{E}_{x\sim D, y\sim\pi_\theta(\cdot|x)}\left[\log\frac{\pi_\theta(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)}\right] \\ &= \mathbb{E}_{x\sim D}\sum_{a_t\in Vocab}\left(\pi_\theta(a_t|s_t)\cdot\log\frac{\pi_\theta(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)}\right),\end{aligned} \tag{42}$$

其中 $\mathcal{Y}$ 为离散类别空间。

A.2 KL 散度梯度推导步骤

步骤 1：展开 KL 表达式 直接写出离散求和形式：

$$\text{KL}\big(\pi_\theta(\cdot|s_t)\|\pi_{\text{ref}}(\cdot|s_t)\big) = \sum_{a_t\in Vocab}\pi_\theta(a_t|s_t)\log\frac{\pi_\theta(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)}. \tag{43}$$

步骤 2：应用梯度算子 对 θ 求梯度：

$$-\nabla_\theta\text{KL}\big(\pi_\theta(\cdot|s_t)\|\pi_{\text{ref}}(\cdot|s_t)\big) = -\nabla_\theta\sum_{a_t\in Vocab}\pi_\theta(a_t|s_t)\log\frac{\pi_\theta(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)}. \tag{44}$$

步骤 3：交换求和与梯度 由于求和项有限且 $\pi_\theta(a_t|s_t)$ 光滑，可交换求和与梯度：

$$-\nabla_\theta\text{KL}\big(\pi_\theta(\cdot|s_t)\|\pi_{\text{ref}}(\cdot|s_t)\big) = -\sum_{a_t\in Vocab}\nabla_\theta\left[\pi_\theta(a_t|s_t)\log\frac{\pi_\theta(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)}\right]. \tag{45}$$

步骤 4：乘积法则分解 对每一项应用乘积法则：

$$\begin{aligned}&-\nabla_\theta\left[\pi_\theta(a_t|s_t)\log\frac{\pi_\theta(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)}\right] \\ &\quad= -[\underbrace{\nabla_\theta\pi_\theta(a_t|s_t)\cdot\log\frac{\pi_\theta(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)}}_{\text{Term 1}} + \underbrace{\pi_\theta(a_t|s_t)\cdot\nabla_\theta\log\frac{\pi_\theta(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)}}_{\text{Term 2}}].\end{aligned} \tag{46}$$

步骤 5：简化 Term 2 由于 $\pi_{\text{ref}}(a_t|s_t)$ 与 θ 无关，有：

$$\nabla_\theta \log \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)} = \nabla_\theta \log \pi_\theta(a_t|s_t) = \frac{\nabla_\theta \pi_\theta(a_t|s_t)}{\pi_\theta(a_t|s_t)}. \tag{47}$$

因此 Term 2 简化为：

$$\pi_\theta(a_t|s_t) \cdot \frac{\nabla_\theta \pi_\theta(a_t|s_t)}{\pi_\theta(a_t|s_t)} = \nabla_\theta \pi_\theta(a_t|s_t). \tag{48}$$

步骤 6：合并两项 将 Term 1 和 Term 2 相加：

$$\sum_{a_t \in Vocab} \left[\nabla_\theta \pi_\theta(a_t|s_t) \cdot \log \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)} + \nabla_\theta \pi_\theta(a_t|s_t) \right]. \tag{49}$$

步骤 7：处理归一化条件 由于 $\sum_{a_t \in Vocab} \pi_\theta(a_t|s_t) = 1$，其梯度为 0：

$$\sum_{a_t \in Vocab} \nabla_\theta \pi_\theta(a_t|s_t) = \nabla_\theta \sum_{a_t \in Vocab} \pi_\theta(a_t|s_t) = \nabla_\theta 1 = 0. \tag{50}$$

因此第二项求和为 0，仅保留第一项：

$$-\nabla_\theta \text{KL}\big(\pi_\theta(\cdot|s_t) \| \pi_{\text{ref}}(\cdot|s_t)\big) = -\sum_{a_t \in Vocab} \nabla_\theta \pi_\theta(a_t|s_t) \cdot \log \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)}. \tag{51}$$

步骤 8：对数导数技巧 利用 $\nabla_\theta \pi_\theta(a_t|s_t) = \pi_\theta(a_t|s_t) \nabla_\theta \log \pi_\theta(a_t|s_t)$，改写为：

$$-\nabla_\theta \text{KL}\big(\pi_\theta(\cdot|s_t) \| \pi_{\text{ref}}(\cdot|s_t)\big) = -\sum_{a_t \in Vocab} \pi_\theta(a_t|s_t) \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot \log \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)}. \tag{52}$$

步骤 9：期望形式 最终梯度可表示为期望：

$$-\nabla_\theta \text{KL}(\pi_\theta(\cdot|s_t) \| \pi_{\text{ref}}(\cdot|s_t)) = -\mathbb{E}_{x \sim D, y \sim \pi_\theta(\cdot|x)} \left[\nabla_\theta \log \pi_\theta(a_t|s_t) \cdot \log \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)} \right]. \tag{53}$$

k2 loss 的损失函数推导 KL 梯度

$$-\nabla_\theta \text{KL}(\pi_\theta(a_t|s_t) \| \pi_{\text{ref}}(a_t|s_t)) \tag{54}$$

$$= -\mathbb{E}_{x \sim D, y \sim \pi_\theta(\cdot|x)} \left[\log \frac{\pi_{\theta_{\text{old}}}(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)} \cdot \nabla_\theta \log \pi_\theta(a_t|s_t) \right] \tag{55}$$

$$
\begin{aligned}
&- \nabla_\theta \mathrm{KL}(\pi_\theta(a_t|s_t) \| \pi_{\text{ref}}(a_t|s_t)) \Big|_{\theta=\theta_{\text{old}}} && (56)\\
&= -\mathbb{E}_{x\sim D, y\sim\pi_{\theta_{\text{old}}}(\cdot|x)} \left[\nabla_\theta \log \pi_{\theta_{\text{old}}}(a_t|s_t) \cdot \log \frac{\pi_{\theta_{\text{old}}}(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)} \right] && (57)\\
&= -\mathbb{E}_{x\sim D, y\sim\pi_{\theta_{\text{old}}}(\cdot|x)} \left[\nabla_\theta \log \pi_\theta(a_t|s_t)|_{\theta=\theta_{\text{old}}} \cdot \log \frac{\pi_{\theta_{\text{old}}}(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)} \right] && (58)\\
&= -\mathbb{E}_{x\sim D, x\sim D, y\sim\pi_{\theta_{\text{old}}}(\cdot|x)} \left[\nabla_\theta \log \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)} \Big|_{\theta=\theta_{\text{old}}} \cdot \log \frac{\pi_{\theta_{\text{old}}}(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)} \right] && (59)\\
&= -\frac{1}{2}\mathbb{E}_{x\sim D, y\sim\pi_{\theta_{\text{old}}}(\cdot|x)} \left[\nabla_\theta \left(\log \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)} \right)^2 \Big|_{\theta=\theta_{\text{old}}} \right] && (60)
\end{aligned}
$$

因此 k_2 loss 下的损失函数

$$
\mathcal{L}_{k_2}(\theta) = \frac{1}{2}\mathbb{E}_{x\sim D, y\sim\pi_{\theta_{\text{old}}}(\cdot|x)} \left[\left(\log \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)} \right)^2 \right] \tag{61}
$$

B DPO 推导

基于 KL 约束的奖励最大化目标，推导出可操作的直接偏好优化目标函数 [7]。首先建立基础优化问题：

$$
\begin{aligned}
&\max_\pi \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi(y|x)}[r(x,y)] - \beta D_{\text{KL}}[\pi(y|x)\|\pi_{\text{ref}}(y|x)]\\
&= \max_\pi \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi(y|x)} \left[r(x,y) - \beta \log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} \right]\\
&= \min_\pi \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi(y|x)} \left[\log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} - \frac{1}{\beta} r(x,y) \right]\\
&= \min_\pi \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi(y|x)} \left[\log \frac{\pi(y|x)}{\frac{1}{Z(x)}\pi_{\text{ref}}(y|x)\exp\left(\frac{1}{\beta} r(x,y)\right)} - \log Z(x) \right]
\end{aligned} \tag{62}
$$

配分函数 $Z(x)$ 构造概率分布：

$$
Z(x) = \sum_{y\in Vocab} \pi_{\text{ref}}(y|x) \exp\left(\frac{1}{\beta} r(x,y)\right) \tag{63}
$$

定义新的参考分布 π^* 为：

$$\pi^*(y|x) = \frac{1}{Z(x)}\pi_{\text{ref}}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right) \tag{64}$$

将目标函数重构为：

$$\begin{aligned}
&\min_{\pi} \mathbb{E}_{x\sim\mathcal{D},y\sim\pi(y|x)}\left[\log\frac{\pi(y|x)}{\pi^*(y|x)} - \log Z(x)\right] \\
&= \min_{\pi} \mathbb{E}_{x\sim\mathcal{D}}\left[\mathbb{E}_{y\sim\pi(y|x)}\left[\log\frac{\pi(y|x)}{\pi^*(y|x)}\right] - \log Z(x)\right] \\
&= \min_{\pi} \mathbb{E}_{x\sim\mathcal{D}}\left[D_{KL}\left(\pi(y|x) \parallel \pi^*(y|x)\right) - \log Z(x)\right]
\end{aligned} \tag{65}$$

由于 $Z(x)$ 与策略 π 无关，优化目标简化为最小化 KL 散度项。根据 KL 散度的非负性，当 $\pi = \pi^*$ 时取得全局最优解。

B.1 从奖励建模到偏好学习

实际应用中直接求解 π^* 存在两大障碍：1) 真实奖励函数 r^* 未知；2) 配分函数 $Z(x)$ 的计算涉及全响应空间积分。为此，我们引入偏好学习框架。

采用 Bradley-Terry 模型，对于输入 x 和响应对 (y_w, y_l)，偏好概率建模为：

$$p^*(y_w \succ y_l|x) = \frac{\exp(r^*(x,y_w))}{\exp(r^*(x,y_w)) + \exp(r^*(x,y_l))} \tag{66}$$

关键突破在于建立奖励函数与最优策略的显式关联。由式 (2) 可得：

$$r(x,y) = \beta\log\frac{\pi^*(y|x)}{\pi_{\text{ref}}(y|x)} + \beta\log Z(x) \tag{67}$$

将奖励差表达式代入偏好概率模型：

$$p^*(y_w \succ y_l|x) = \frac{1}{1+\exp\left(\beta\log\frac{\pi^*(y_l|x)}{\pi_{\text{ref}}(y_l|x)} - \beta\log\frac{\pi^*(y_w|x)}{\pi_{\text{ref}}(y_w|x)}\right)} \tag{68}$$

B.2 最终目标函数

通过极大似然估计，得到直接优化策略的参数化目标函数：

$$\mathcal{L}_{\text{DPO}}(\pi_\theta;\pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta\log\frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right] \tag{69}$$

B.3 DPO 梯度

$$\nabla_\theta \mathcal{L}_{\mathrm{DPO}}(\pi_\theta; \pi_{\mathrm{ref}}) =$$

$$-\beta \mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \left[\underbrace{\sigma(\hat{r}_\theta(x, y_l) - \hat{r}_\theta(x, y_w))}_{\text{higher weight when reward estimate is wrong}} \left[\underbrace{\nabla_\theta \log \pi(y_w \mid x)}_{\text{increase likelihood of } y_w} - \underbrace{\nabla_\theta \log \pi(y_l \mid x)}_{\text{decrease likelihood of } y_l} \right] \right]. \tag{70}$$

C 重要性采样与 PPO-2 clip

C.1 基本概念与动机

重要性采样（Importance Sampling）是强化学习中实现离策略（off-policy）学习的关键技术, 其核心思想是通过引入行为策略（behavior policy）的采样分布来估计目标策略（target policy）的期望值。这一方法在策略优化中具有双重意义：

1. 样本复用：允许利用历史策略生成的旧样本进行当前策略更新，显著提升数据利用率 2. 方差控制：通过重要性权重修正新旧策略的概率分布偏差，维持无偏估计特性

C.2 策略梯度推导

考虑策略梯度基本形式：

$$\nabla_\theta J(\theta) = \mathbb{E}_{(a_t,s_t)\sim\pi_\theta} \left[A(x, y) \nabla_\theta \log \pi_\theta(a_t|s_t) \right] \tag{71}$$

当转换为离策略更新时，需要引入重要性权重（Importance Weight）$\rho_t = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)}$：

$$\begin{aligned}
\nabla_\theta J(\theta) &= \mathbb{E}_{(a_t,s_t)\sim\pi_{\theta'}} \left[\frac{\pi_\theta(a_t, s_t)}{\pi_{\theta'}(a_t, s_t)} A_\theta(a_t, s_t) \nabla_\theta \log \pi_\theta(a_t|s_t) \right] \\
&= \mathbb{E}_{(a_t,s_t)\sim\pi_{\theta'}} \left[\frac{\pi_\theta(a_t|s_t)\cancel{\pi_\theta(s_t)}}{\pi_{\theta'}(a_t|s_t)\cancel{\pi_{\theta'}(s_t)}} A_\theta(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t|s_t) \right] \\
&\approx \mathbb{E}_{(a_t,s_t)\sim\pi_{\theta'}} \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta'}(a_t|s_t)} A_{\theta'}(a_t, s_t) \nabla_\theta \log \pi_\theta(a_t|s_t) \right]
\end{aligned} \tag{72}$$

推导过程中包含两个重要近似：

- 状态分布抵消假设：$\pi_\theta(s_t) \approx \pi_{\theta'}(s_t)$，在策略更新幅度较小时成立
- 优势函数近似：$A_\theta(a_t, s_t) \approx A_{\theta'}(a_t, s_t)$，要求新旧策略差异可控

C.3 目标函数形式化

令 $\theta' = \theta_{\text{old}}$，得到离策略目标函数：

$$J(\theta) = \mathbb{E}_{x\sim D, y\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} A_{\theta_{\text{old}}}(a_t, s_t)\right] \tag{73}$$

该目标函数具有以下特性：

1. **无偏性**：当 π_θ 与 $\pi_{\theta_{\text{old}}}$ 的支撑集相同时保持无偏估计
2. **方差敏感性**：重要性权重 ρ_t 的数值稳定性直接影响梯度质量
3. **策略约束**：需通过 KL 散度等度量限制 π_θ 与 $\pi_{\theta_{\text{old}}}$ 的差异

C.4 PPO-2 clip 方法

针对重要性采样方差问题，PPO-2 clip 提出带截断的替代目标函数：

$$J^{\text{clip}}(\theta) = \mathbb{E}_t\left[\min\left(\rho_t A_{\theta_{\text{old}}}(a_t, s_t),\ \text{clip}(\rho_t, 1-\epsilon, 1+\epsilon) A_{\theta_{\text{old}}}(a_t, s_t)\right)\right] \tag{74}$$

其中 $\rho_t = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$，$\epsilon$ 为截断阈值（通常取 0.1-0.2）。该方法通过双向截断实现：

1. 当 $A_t > 0$ 时，限制策略提升幅度不超过 $1+\epsilon$ 2. 当 $A_t < 0$ 时，限制策略下降幅度不超过 $1-\epsilon$

核心优势：

- **梯度稳定**：通过硬性截断避免过大策略更新
- **计算高效**：无需计算 KL 散度等附加约束项
- **兼容优势**：兼容 GAE 等多种优势估计方法

该设计在保证样本效率的同时，显著提升了策略优化的稳定性，成为深度强化学习领域最广泛应用的算法之一。

D k3 估计不存在绝对优势

Schulman, John [8] 认为 k3 估计相比 k1、k2 估计存在绝对优势，有着更小的偏差和方差。但即便是在纯 KL 散度估计领域，其实也并不成立。相比原始实现 [8]，下面代码对 truekl 的计算进行了修正：

```python
import torch.distributions as dis
import torch
p = dis.Normal(loc=0, scale=1)
q = dis.Normal(loc=0.1, scale=0.2)
x = q.sample(sample_shape=(10_000,))
truekl = dis.kl_divergence(q, p)
print("true", truekl)
logr = p.log_prob(x) - q.log_prob(x)
k1 = -logr
k2 = logr ** 2 / 2
k3 = (logr.exp() - 1) - logr
k4 = torch.where(k1 < 0, k3, torch.minimum(k1, k3))
r = torch.exp(logr)
clip_r = torch.clamp(r,max=10)
k3_clip = clip_r - 1 - logr
for k in (k1, k2, k3, k4, k3_clip):
    print((k.mean() - truekl) / truekl, k.std() / truekl)
```

运行结果为：

```
truekl tensor(1.1344)
tensor(0.0007) tensor(0.5896)
tensor(-0.2348) tensor(0.4524)
tensor(-0.4051) tensor(2.4417)
tensor(-0.4051) tensor(2.4417)
tensor(-0.4676) tensor(0.4370)
```

k3 估计的方差是远大于 k1 估计和真实值的，这主要源于它进行了 exp() 计算，为此我们提出 k3_clip，限制 exp() 的最大值，避免数值非线性增长带来的问题。

E GROUP Norm 方差一定 <1

当 reward 列表中的元素限制在 0 到 1 之间时，其方差必定不大于 1。以下是对这个结论的严谨数学证明:

数学证明 设列表为 $x_1, x_2, \cdots, x_n$，其中每个元素满足 $0 \leq x_i \leq 1$。

列表的均值定义为：$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$

方差的计算公式为：$\mathrm{Var}(X) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$

推理过程如下：

1. 由于每个元素满足 $0 \leq x_i \leq 1$，因此均值也必然在此范围内，即 $0 \leq \bar{x} \leq 1$。

2. 对于任意元素 x_i，其与均值的差值范围为 $-1 \leq x_i - \bar{x} \leq 1$。- 当 $x_i = 0$ 且 $\bar{x} = 1$ 时，达到最小值 -1 - 当 $x_i = 1$ 且 $\bar{x} = 0$ 时，达到最大值 1

3. 对差值平方后，得到 $0 \leq (x_i - \bar{x})^2 \leq 1$。

4. 由于方差 $\mathrm{Var}(X)$ 是这些平方差值的平均值，因此方差也必然满足：$0 \leq \mathrm{Var}(X) \leq 1$。

F 采样 KL 梯度系数可视化代码

```python
import torch
import matplotlib.pyplot as plt

# 设定一个范围用于绘制曲线
pertokenlogps = torch.linspace(-3, 2, steps=400)

# 计算每条曲线的值
curve2 = pertokenlogps + 0.7
curve3 = 1 - torch.exp(-0.7 - pertokenlogps)
curve4 = torch.exp(pertokenlogps) - torch.exp(torch.tensor(-0.7))
# 绘制曲线
plt.figure(figsize=(10, 6))
plt.plot(pertokenlogps, curve2, label='pertokenlogps - (-0.7)  (k1/k2)')
plt.plot(pertokenlogps, curve3, label='1 - exp(-0.7 - pertokenlogps)   (k3)')
plt.plot(pertokenlogps, curve4, label='exp(pertokenlogps) - exp(-0.7) minimax')
plt.xlabel('pertokenlogps')
plt.ylabel('Value')
plt.title('set log \pi_ref = -0.7, as log(0.5) = -0.69.')
plt.legend()
plt.grid(True)
plt.show()
```